بسم الله الرحمن الرحيم

Backpropagation Neural Network Architecture

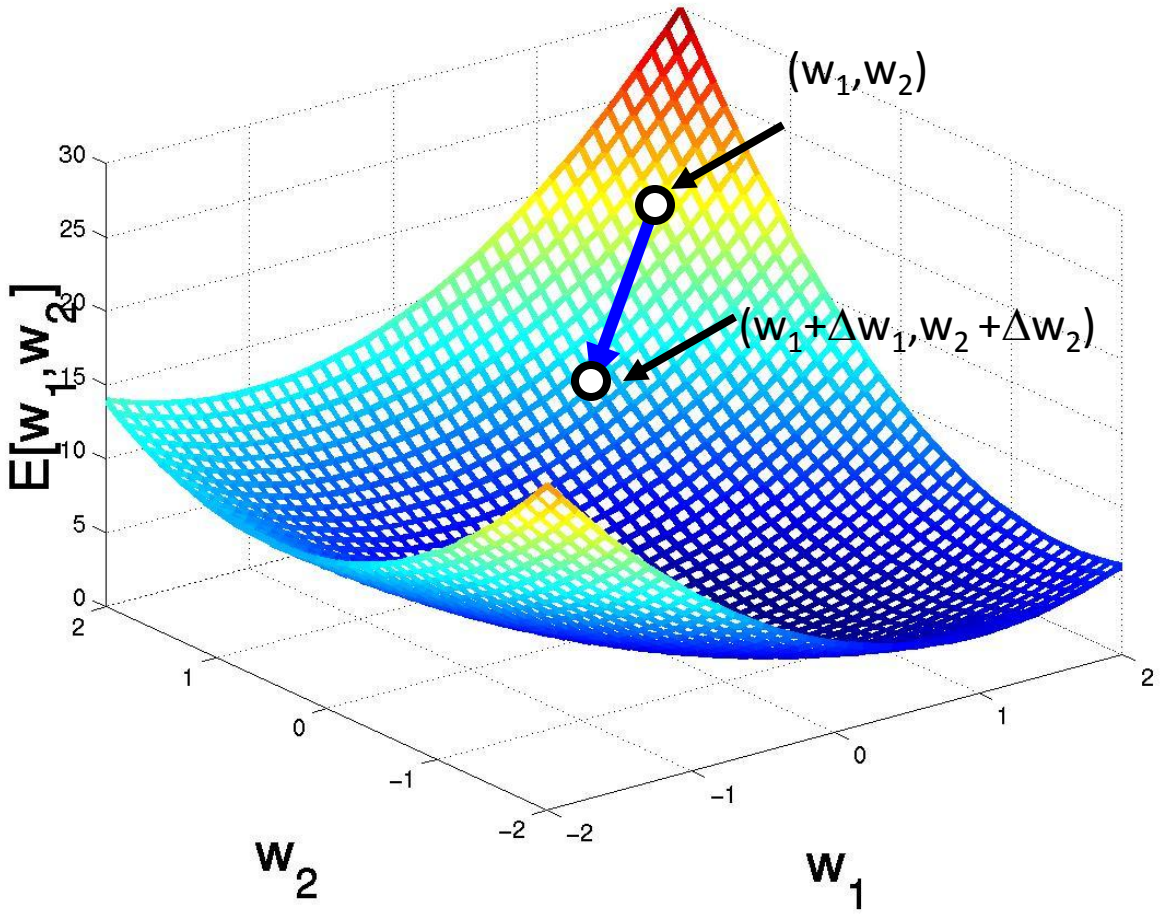
The BP network implements the generalized delta rule. It is a gradient descent algorithm which minimizes the squared error of the network. The gradient descent algorithm is applied to adjust the connected weights.

The training process of the BP neural network generally involves five steps:

1. Select representative training samples and turn them into the input layer as the input value.
2. Calculate the predictive value of the network.
3. Compare the target value with the predictive value to obtain the error value.
4. Readjust the weights in each layer of the network according to the error value.

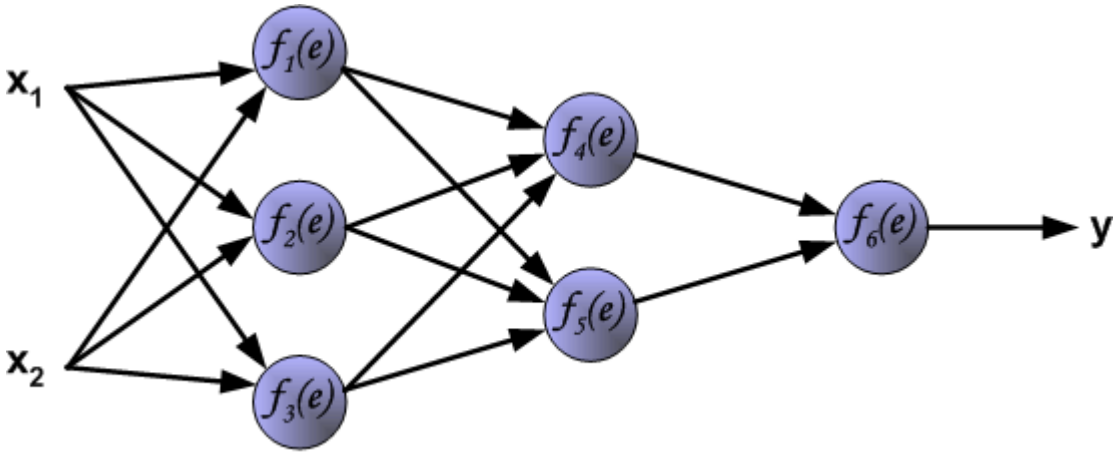
Repeat the above procedure until the error value of each training sample is minimized, meaning that the training is finished.

Gradient Descent



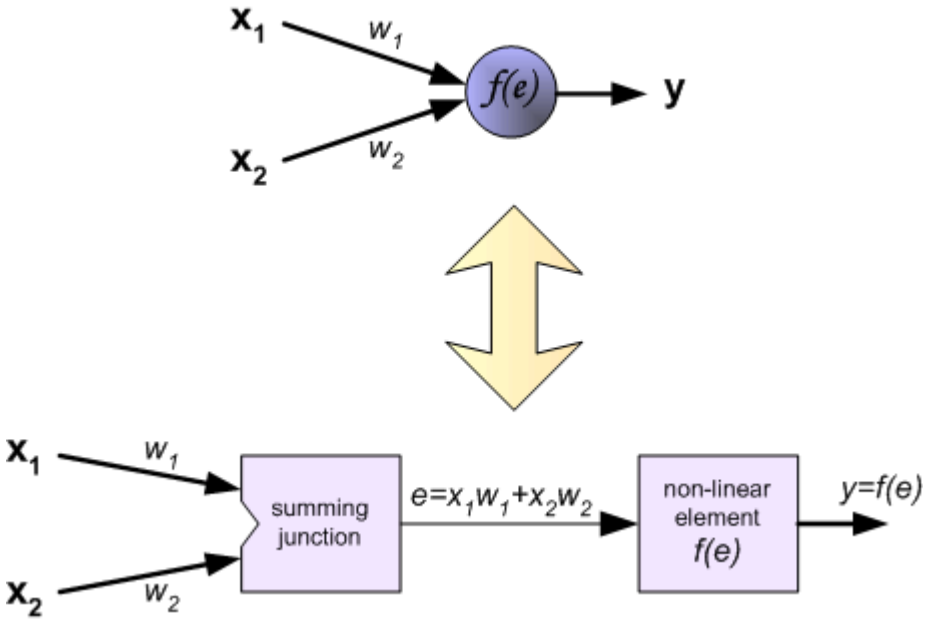
Learning in Artificial Neural Networks

Backpropagation Neural Network



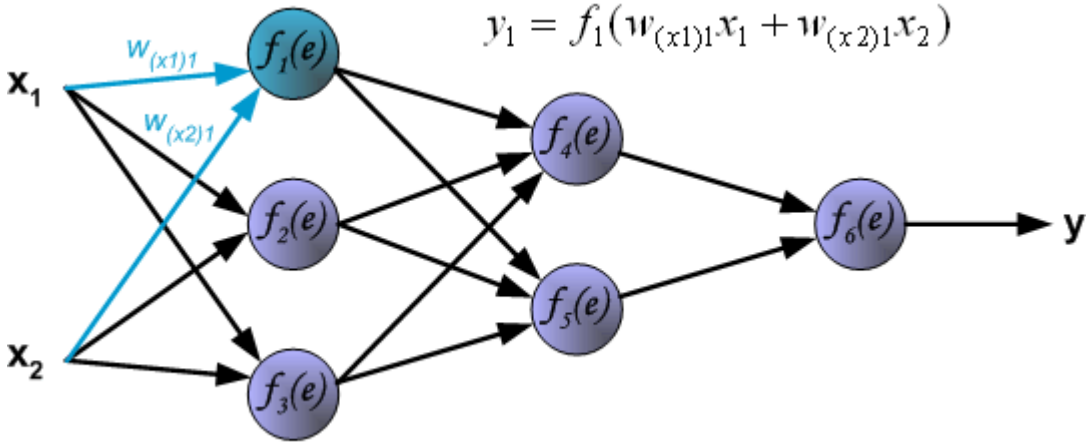
Learning in Artificial Neural Networks

Backpropagation Neural Network



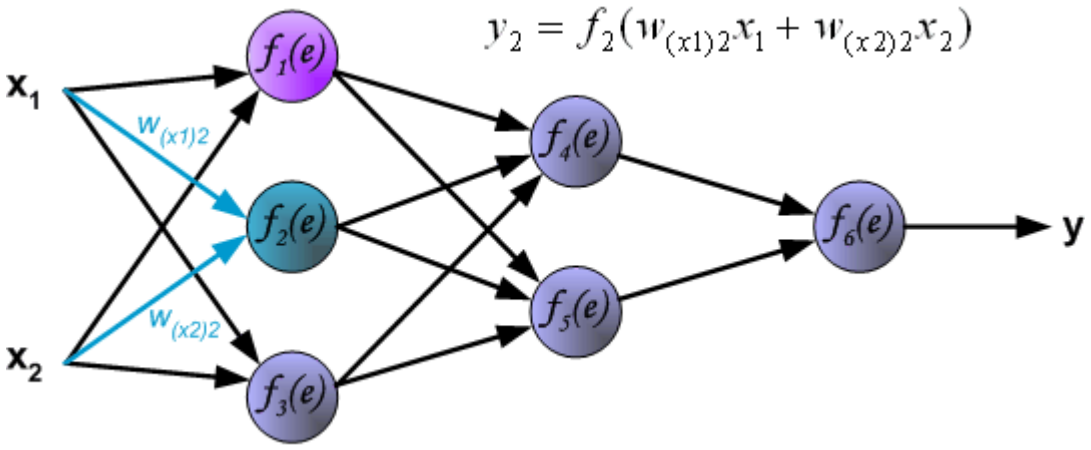
Learning in Artificial Neural Networks

Backpropagation Neural Network



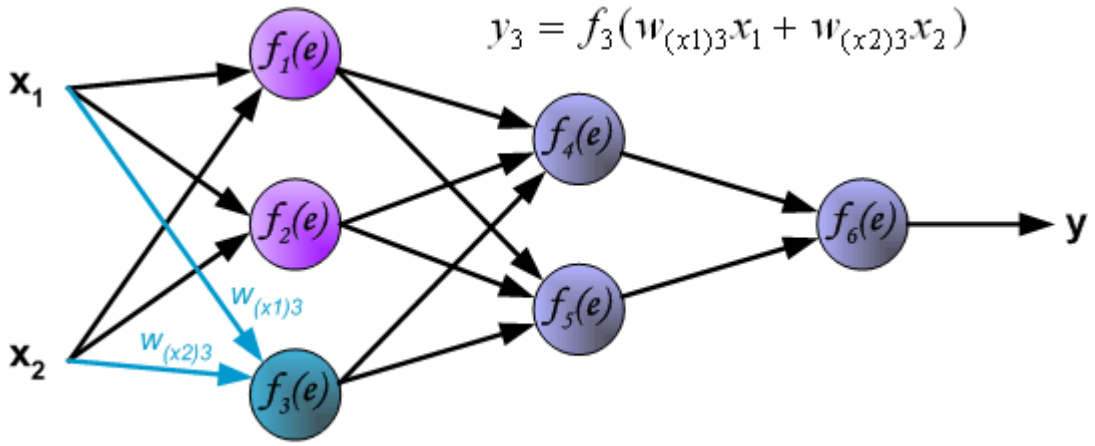
Learning in Artificial Neural Networks

Backpropagation Neural Network



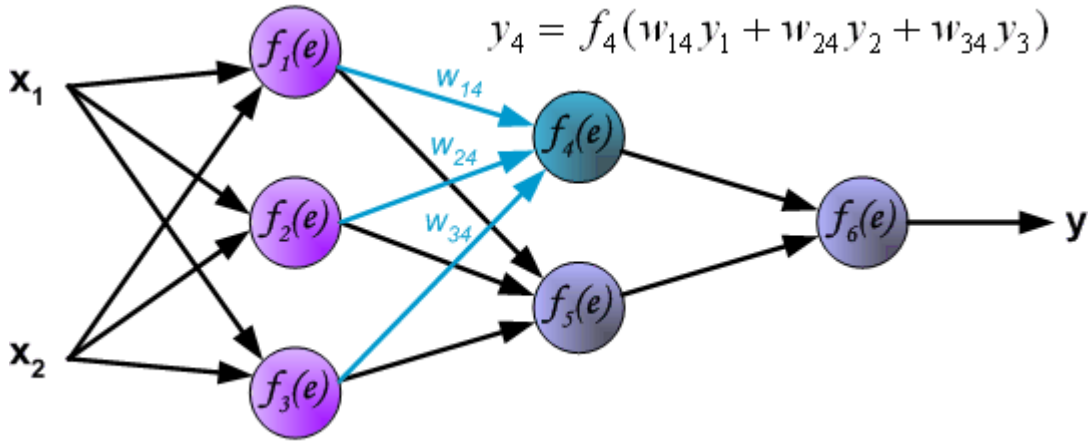
Learning in Artificial Neural Networks

Backpropagation Neural Network



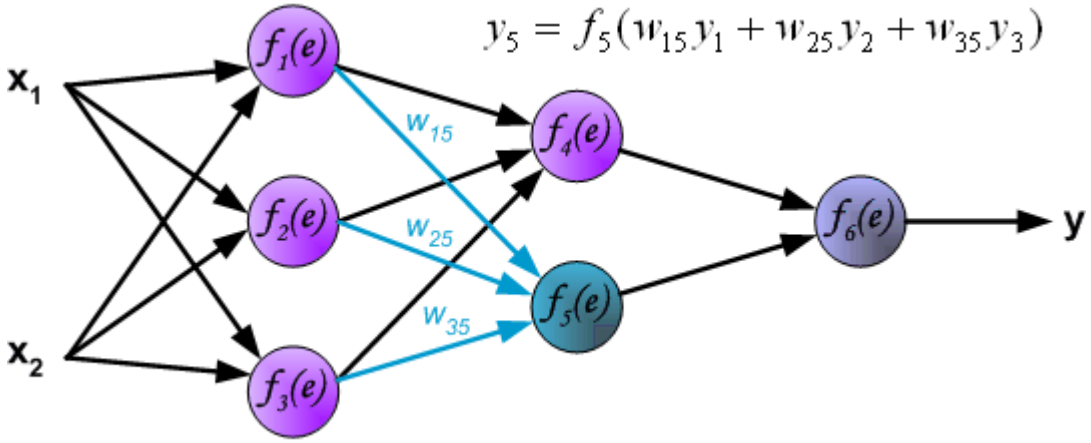
Learning in Artificial Neural Networks

Backpropagation Neural Network



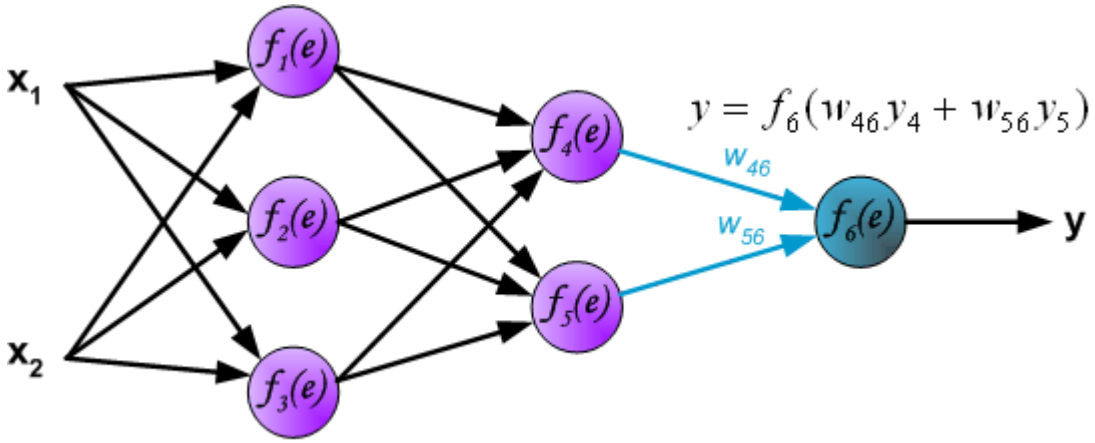
Learning in Artificial Neural Networks

Backpropagation Neural Network



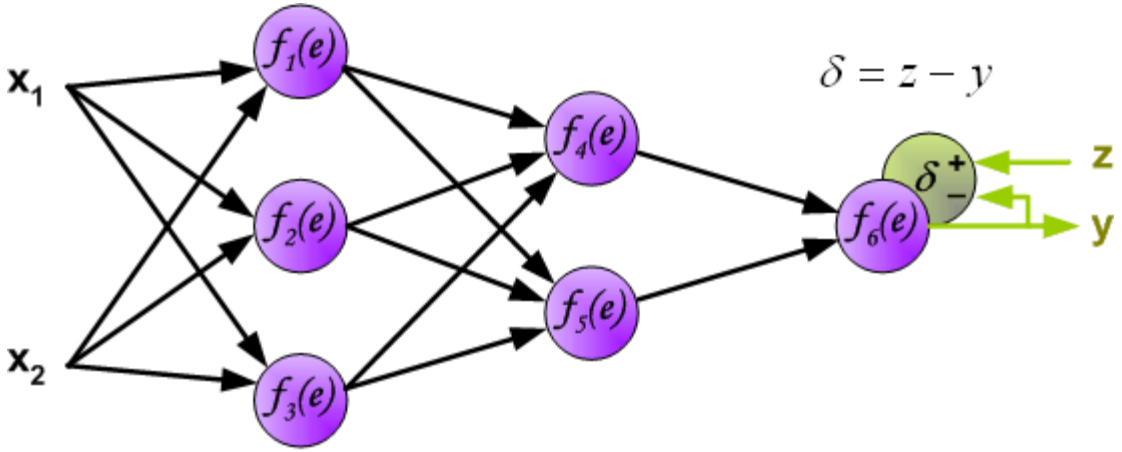
Learning in Artificial Neural Networks

Backpropagation Neural Network



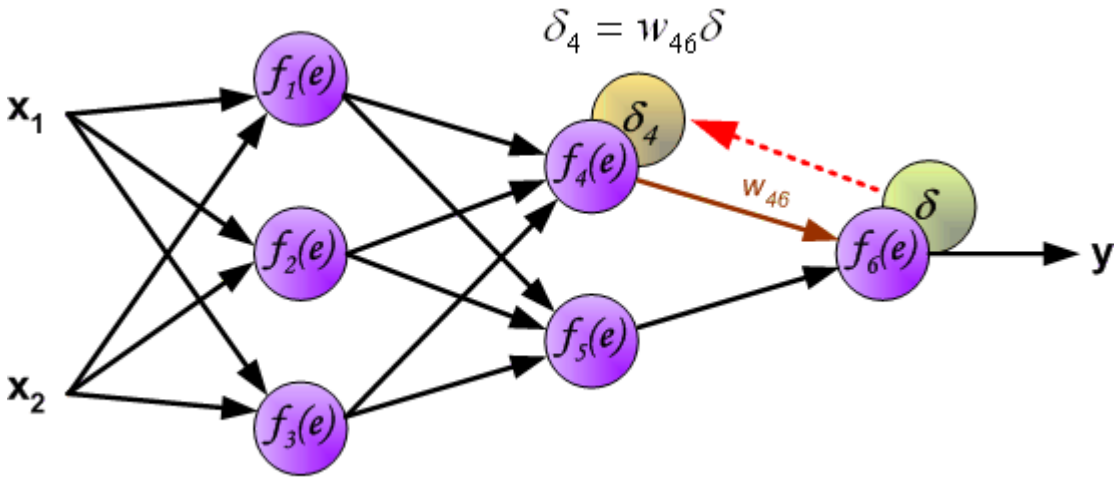
Learning in Artificial Neural Networks

Backpropagation Neural Network



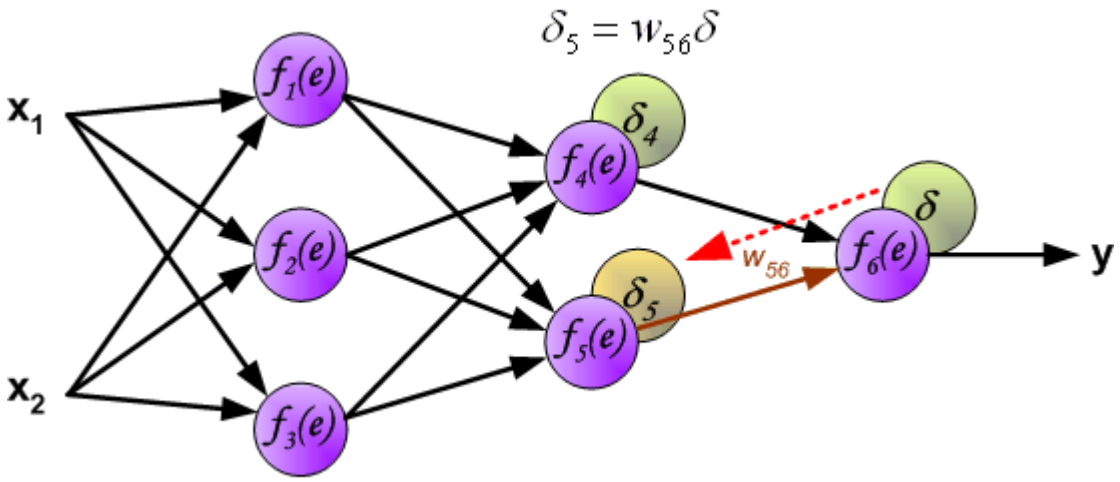
Learning in Artificial Neural Networks

Backpropagation Neural Network



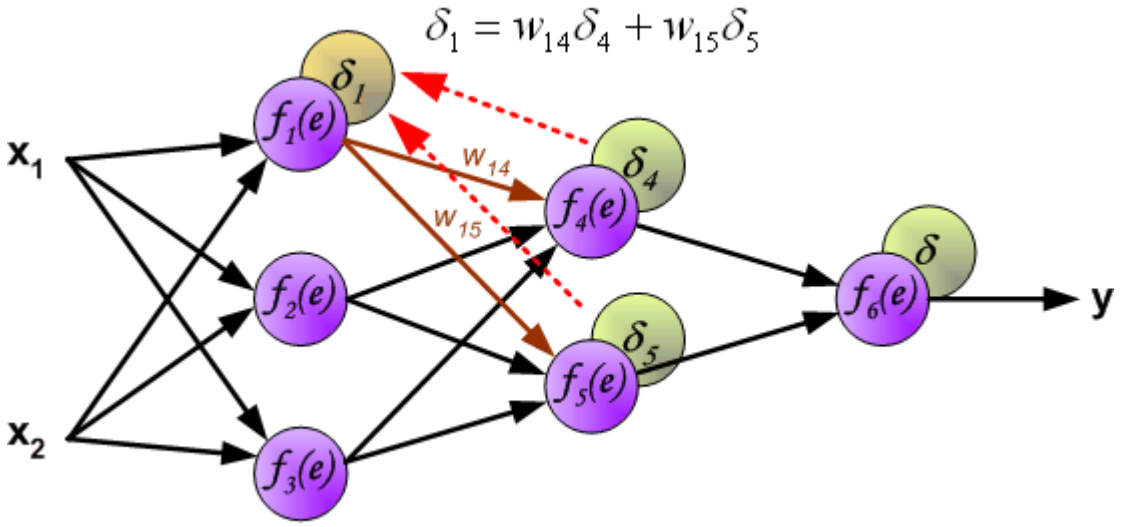
Learning in Artificial Neural Networks

Backpropagation Neural Network



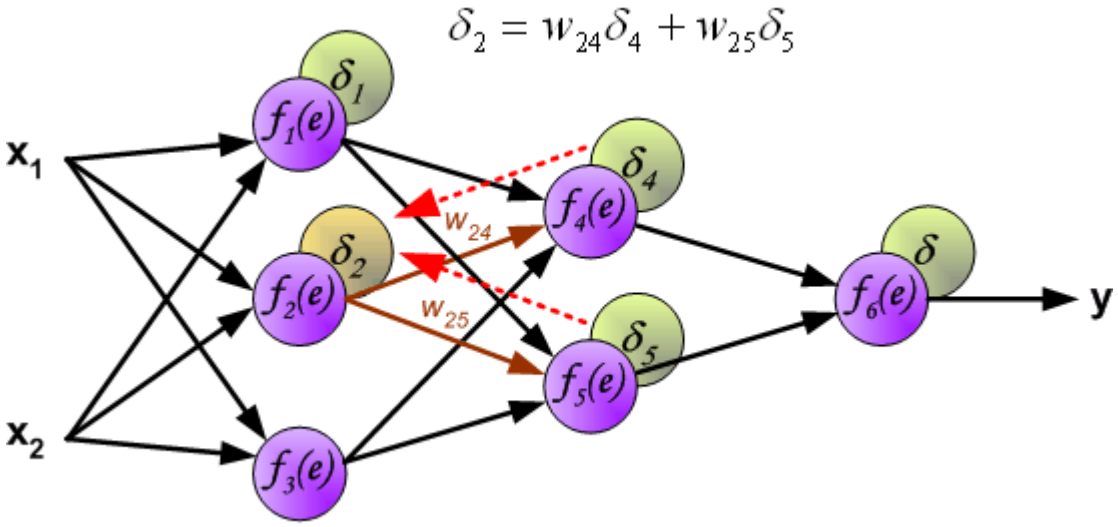
Learning in Artificial Neural Networks

Backpropagation Neural Network



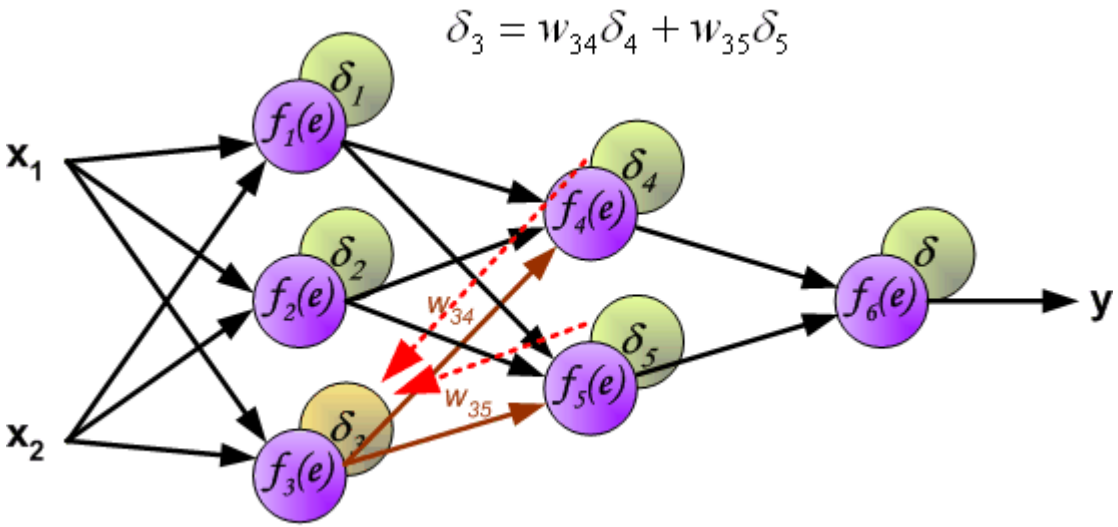
Learning in Artificial Neural Networks

Backpropagation Neural Network



Learning in Artificial Neural Networks

Backpropagation Neural Network

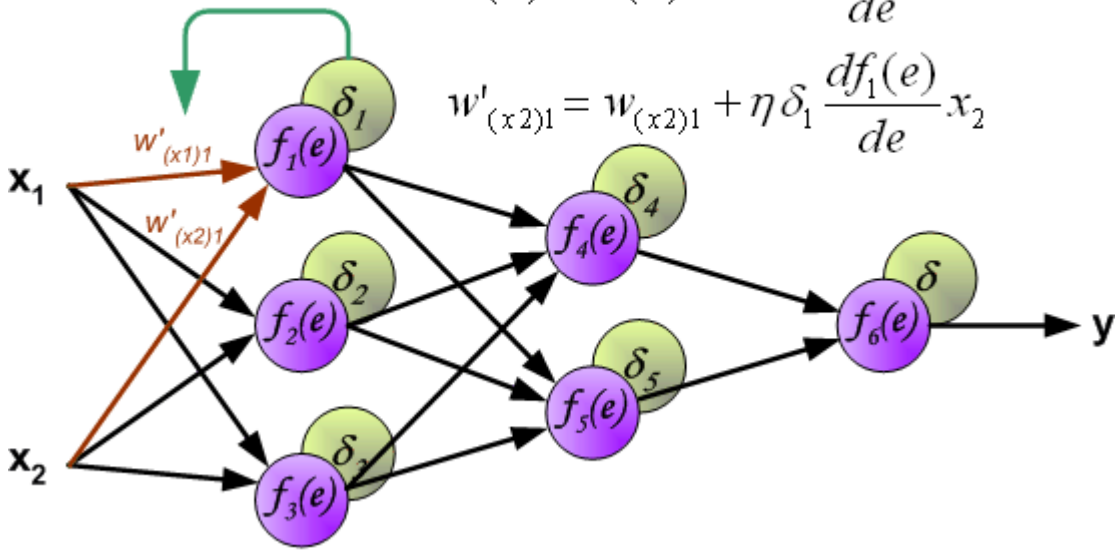


Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

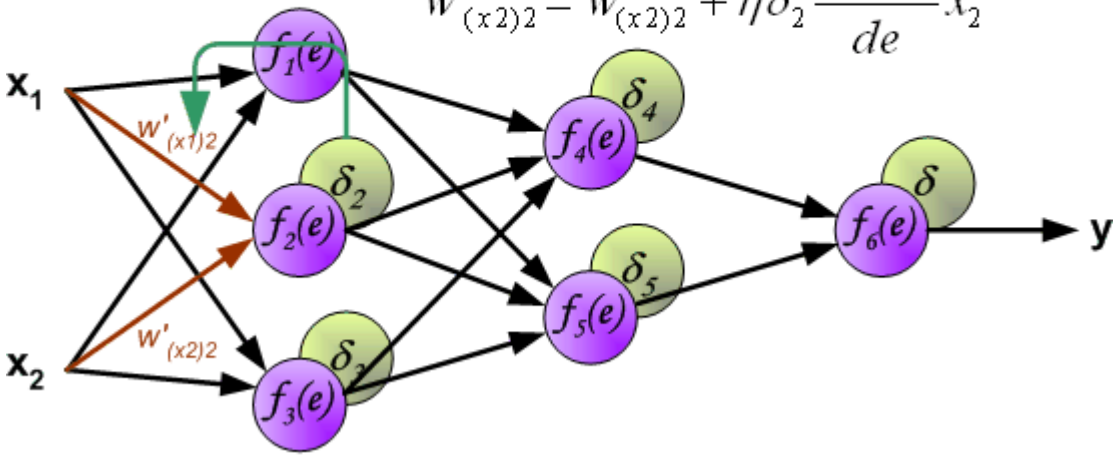


Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

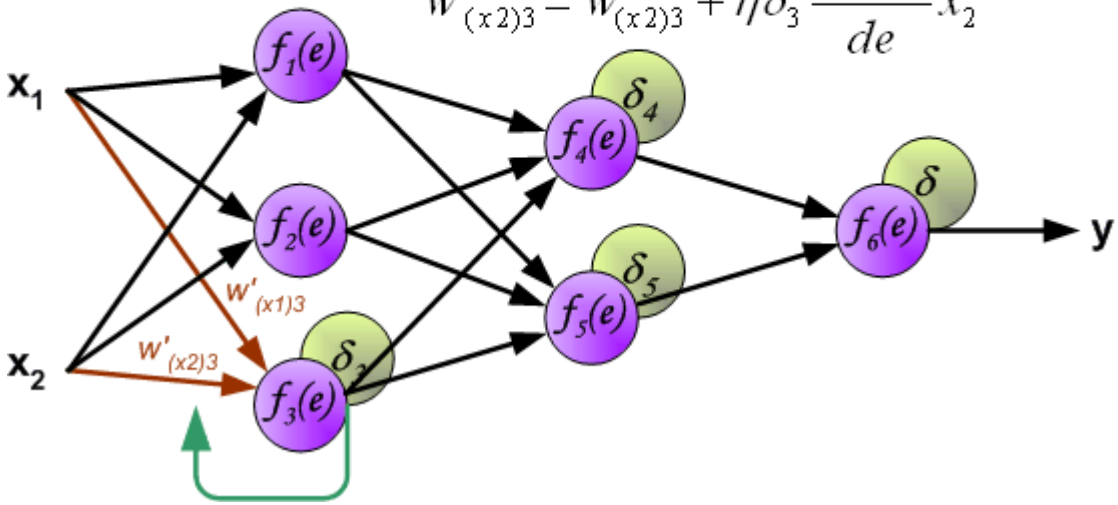


Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$



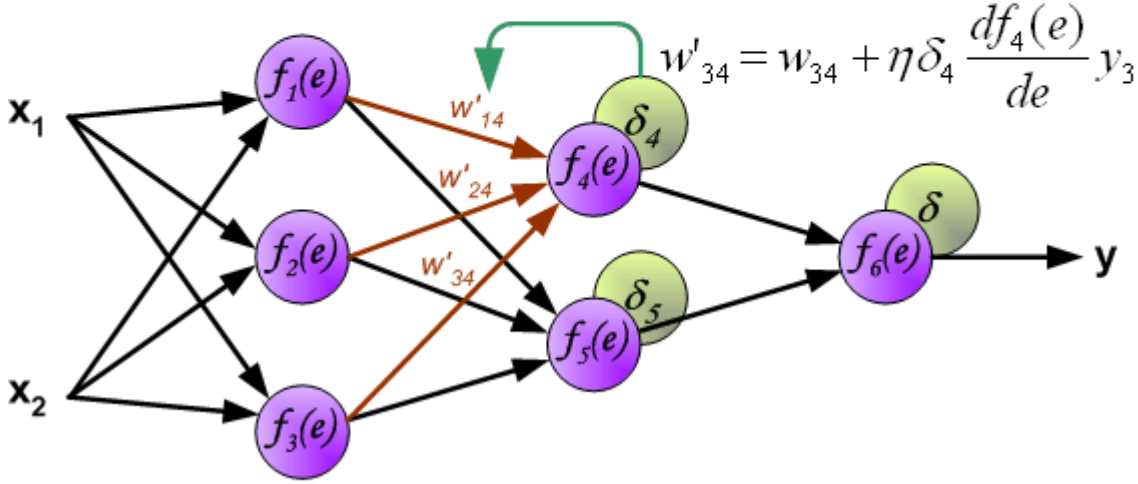
Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



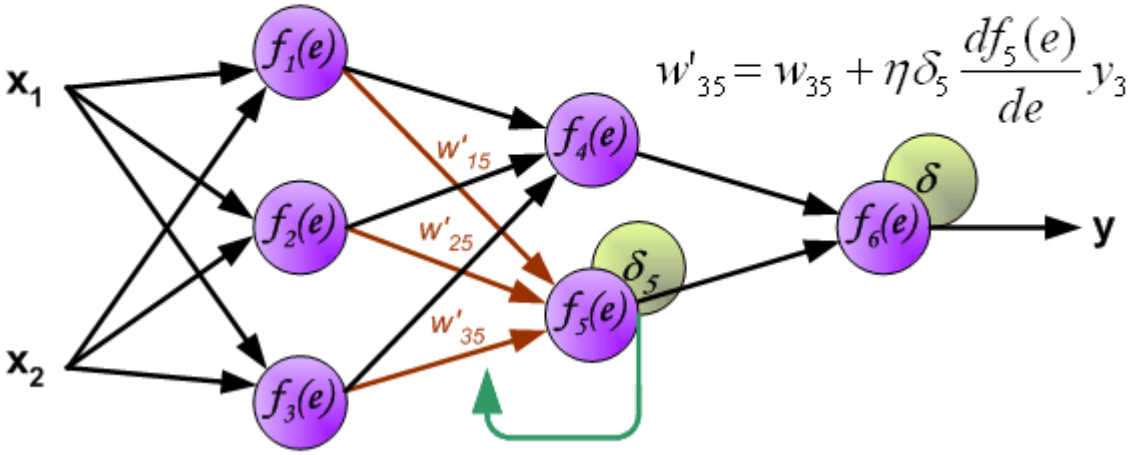
Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$

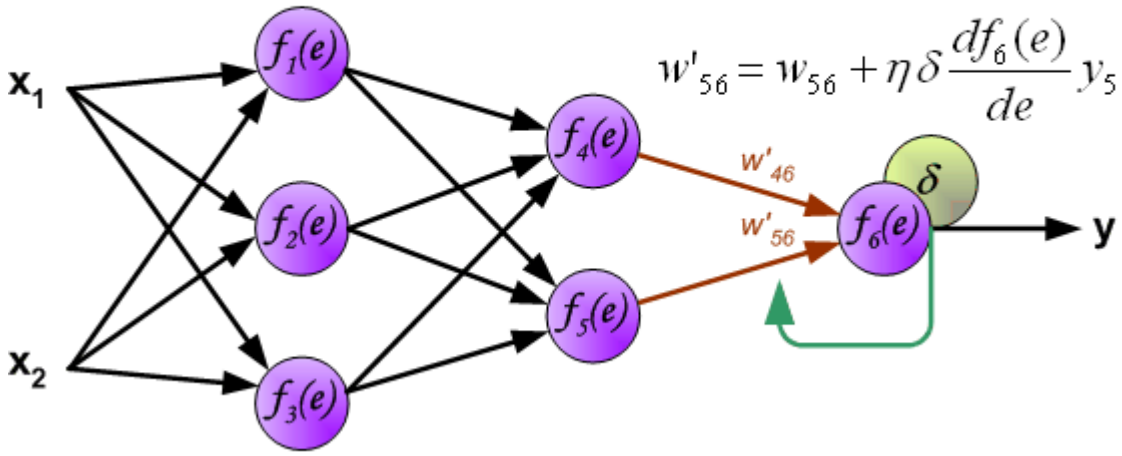


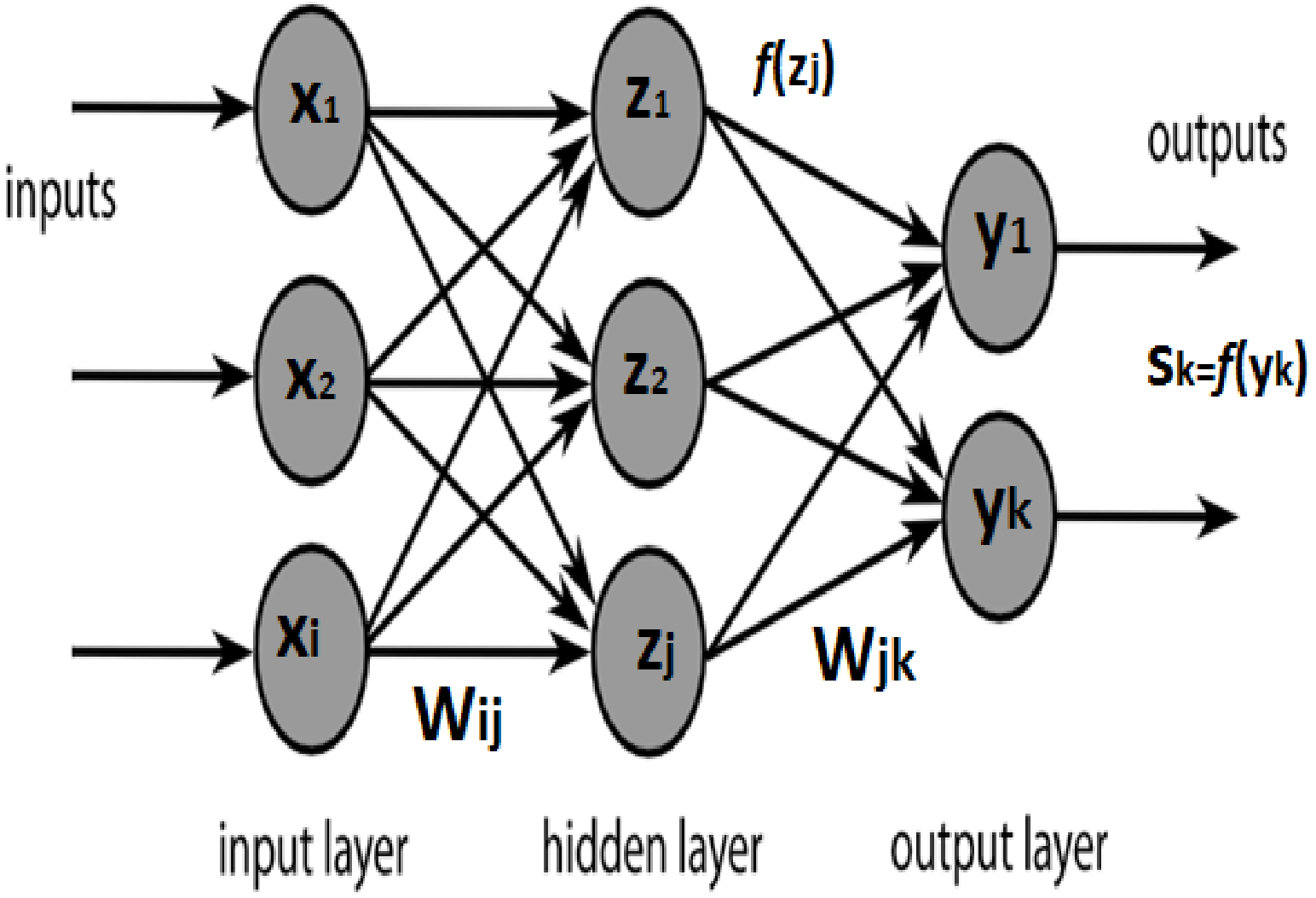
Learning in Artificial Neural Networks

Backpropagation Neural Network

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$




inputs
x1
x2
xi
z1
z2
zj
f(zj)
y1
yk
outputs
Sk=f(yk)
Wij
Wjk
input layer
hidden layer
output layer

The following is the scenario:

The output of the *k*th output neuron, s_k is:

$$s_k = f(y_k)$$

where y_k is the network input to the *k*th output neuron which is written as:

$$y_k = \sum_j f(z_j)\, w_{jk}$$

where $f(z_j)$ is the output of the *j*th hidden unit

where z_j is the network input to the *j*th hidden neuron which is written as:

$$z_j = \sum_i x_i w_{ij}$$

where x_i is *i*th input unit and w_{ij} is the weights of the hidden layer.

The error function to be minimized is:

$$E = \frac{1}{2}\sum_{k=1}^{n}(d - f(y_k))^2$$

where *d* is the desired output vector.

for weight connected from hidden to output units:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial f(y_k)} \frac{\partial f(y_k)}{\partial y_k} \frac{\partial y_k}{\partial w_{jk}}$$

where w_{jk} is the weight of the output layer.

$$\boldsymbol{let\ f(y_k) = sigmoidal\ function}$$

$$\frac{\partial E}{\partial w_{jk}} = -(d - f(y_k)) f(y_k)(1 - f(y_k)) \frac{\partial y_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -\, e\, f(y_k)\big(1 - f(y_k)\big)\ f(z_j)$$

Let $\delta_k = e\, f(y_k)(1 - f(y_k))$

$$\frac{\partial E}{\partial w_{jk}} = -\, \delta_k\, f(z_j)$$

for weight connected from input to hidden units:

$$\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k \frac{\partial E}{\partial f(y_k)} \frac{\partial f(y_k)}{\partial y_k} \frac{\partial y_k}{\partial f(z_j)}\right) \frac{\partial f(z_j)}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k -e \; f(y_k)(1 - f(y_k))\, w_{jk}\right) f(z_j)(1 - f(z_j)) \; x_i$$

$$\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k \delta_k \; w_{jk}\right) f(z_j)(1 - f(z_j)) \; x_i$$

Let $$\delta_j = \left(\sum_k \delta_k \; w_{jk}\right) f(z_j)(1 - f(z_j))$$

$$\frac{\partial E}{\partial w_{ij}} = -\,\delta_j \; x_i$$

the change in weights of the output units are given by

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

$$= \eta \delta_k f(z_j)$$

where η learning rate parameter.

The change in weights of the hidden units is

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

$$= \eta \, \delta_j \, x_i$$

The weights update equations are given in

$$w_{jk}{}^{new} = w_{jk}{}^{old} + \Delta w_{jk}$$

$$w_{ij}{}^{new} = w_{ij}{}^{old} + \Delta w_{ij}$$

where $w_{ji}{}^{new}$ and $w_{ji}{}^{new}$ are the new weights, $w_{ji}{}^{old}$ and $w_{ji}{}^{old}$ are the previous weights.